Package ‘scs’

November 19, 2019

Version 1.3-2
Title Splitting Conic Solver
Description Solves convex cone programs via operator splitting. Can solve:
linear programs (‘LPs’), second-order cone programs (‘SOCPs’), semidefinite programs
(‘SDPs’), exponential cone programs (‘ECPs’), and power cone programs (‘PCPs’), or
problems with any combination of those cones. ‘SCS’ uses ‘AMD’ (a set of routines for permut-
ing sparse matrices prior to factorization) and ‘LDL’ (a sparse ‘LDL’ factorization and solve pack-
age) from ‘SuiteSparse’ (<http://www.suitesparse.com>).
Depends R (>= 2.15)
SystemRequirements GNU Make
Suggests slam, testthat
Encoding UTF-8
License GPL-3
URL https://github.com/FlorianSchwendinger/scs
RoxygenNote 6.1.1
NeedsCompilation yes
Author Florian Schwendinger [ctb, cre],
Brendan O’Donoghue [aut, cph],
Balasubramanian Narasimhan [ctb],
Timothy A. Davis [cph],
Patrick R. Amestory [cph],
Iain S. Duff [cph]
Maintainer Florian Schwendinger <FlorianSchwendinger@gmx.at>
Repository CRAN
Date/Publication 2019-11-19 18:40:02 UTC

R topics documented:

  scs .................................................. 2
  scs_control ........................................ 3

Index 5
SCS - Splitting Conic Solver

Description

Solves convex cone programs via operator splitting.

Usage

scs(A, b, obj, cone, control = scs_control())

Arguments

- **A**: a matrix of constraint coefficients. **NOTE**: The rows of matrix A have to be ordered according to the order given in subsection “Allowed cone parameters”. For more information see README.
- **b**: a numeric vector giving the primal constraints
- **obj**: a numeric vector giving the primal objective
- **cone**: a list giving the cone sizes
- **control**: a list giving the control parameters. For more information see README.

Details

**Important Note:**
The order of the rows in matrix A has to correspond to the order given in the table “Cone Arguments”, which means rows corresponding to primal zero cones should be first, rows corresponding to non-negative cones second, rows corresponding to second-order cone third, rows corresponding to positive semidefinite cones fourth, rows corresponding to exponential cones fifth and rows corresponding to power cones at last.

**SCS can solve:**
1. linear programs (LPs)
2. second-order cone programs (SOCPs)
3. semidefinite programs (SDPs)
4. exponential cone programs (ECPs)
5. power cone programs (PCPs)
6. problems with any combination of cones, which can be defined by the parameters listed in the subsection “Allowed cone parameters”

**Allowed cone parameters are:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>integer</td>
<td>1</td>
<td>number of primal zero cones (dual free cones), which corresponds to the primal equality constraints</td>
</tr>
<tr>
<td>l</td>
<td>integer</td>
<td>1</td>
<td>number of linear cones (non-negative cones)</td>
</tr>
</tbody>
</table>
**scs_control**

```r
c(s(1, 1), ncol=1)
b <- c(1, 1)
obj <- 1
cone <- list(f = 2)
control <- list(eps = 1e-3, max_iters = 50)
sol <- scs(A, b, obj, cone, control)
sol
```

---

**Description**

Details to the `control` parameters.

**Usage**

```r
scs_control(max_iters = 5000L, normalize = TRUE, verbose = FALSE,
cg_rate = 2, scale = 1, rho_x = 0.001, alpha = 1.5,
eps = 1e-05, acceleration_lookback = 10L)
```

**Arguments**

- `max_iters` an integer giving the maximum number of iterations (default is 5000L).
- `normalize` a logical giving if heuristic data rescaling should be used (default is TRUE).
- `verbose` a logical giving if the progress should be printed (default is FALSE).
- `cg_rate` a double giving the rate at which the CG tolerance for the indirect method is tightened (higher is tighter, default is 2.0).
- `scale` a double giving the factor (default is 1.0) by which the data is rescaled (only used if normalize is TRUE).
- `rho_x` a double giving the momentum of x term (default is 1e-3).
- `alpha` a double giving the over-relaxation parameter, allowed values are in (0, 2) (default if 1.5).
- `eps` a double giving the convergence tolerance (default is 1e-5).
acceleration_lookback

an integer indicating the number of iterations to look back for Anderson acceleration (default is 10L).

Value

a list containing the control parameters.
Index

scs, 2
scs_control, 3